On the Parameterized Complexity of the Structure of Lineal Topologies (Depth-First Spanning Trees) of Finite Graphs: The Number of Leaves.*

Emmanuel Sam[1][0000−0001−7756−0901], Michael Fellows[1,2][0000−0002−6148−9212], Frances Rosamond[1,2][0000−0002−5097−9929], and Petr A. Golovach[1][0000−0002−2619−2990]

[1] Department of Informatics, University of Bergen, Norway
[2] Western Sydney University, Locked Bag 1797, Penrith NSW 2751, Australia
{emmanuel.sam,michael.fellows,frances.rosamond,petr.golovach}@uib.no

Abstract. A *lineal topology* $\mathcal{T} = (G, r, T)$ of a graph G is an r-rooted depth-first spanning (DFS) tree T of G. Equivalently, this is a spanning tree of G such that every edge uv of G is either an edge of T or is between a vertex u and an *ancestor* v on the unique path in T from u to r. We consider the parameterized complexity of finding a lineal topology that satisfies upper or lower bounds on the number of leaves of T, parameterized by the bound. This immediately yields four natural parameterized problems: (i) $\leq k$ leaves, (ii)$\geq k$ leaves, (iii) $\leq n-k$ leaves, and (iv) $\geq n-k$ leaves, where $n = |G|$. We show that all four problems are NP-hard, considered classically. We show that (i) is para-NP-hard, (ii) is hard for W[1], (iii) is FPT, and (iv) is FPT. Our work is motivated by possible applications in graph drawing and visualization.

Keywords: DFS tree · Spanning tree · Parameterized complexity.

1 Introduction

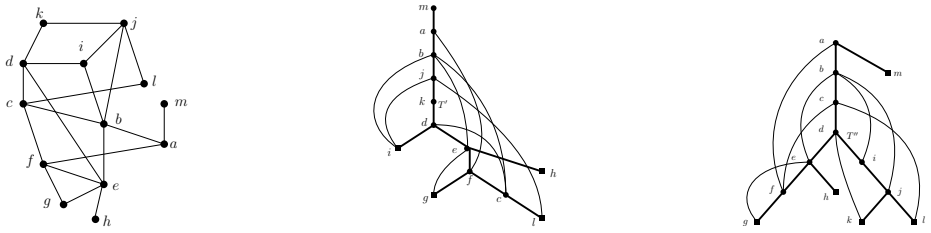
For every connected undirected graph $G = (V, E)$ with vertex set $V(G)$ and edge set $E(G)$, there exists a rooted spanning tree T having the property that for every edge $xy \in E(G)$ that is not an edge of T, either x is a descendant of y with respect to T, or x is an ancestor of y. Such a tree is called a *depth-first spanning tree* (or *DFS-tree* for short), as one may be computed by depth–first search (DFS), and the edges of G that are not part of T are referred to as *back edges* [21]. It has also been called *lineal spanning tree* [29], *trémaux tree* [10], and *normal spanning tree*, particularly in the case of infinite graphs [12].

The importance of the properties of such trees in the design of efficient algorithms is evident in the great variety of algorithms that employ DFS to solve graph-theoretic problems, including finding connected and biconnected components of undirected graphs [21], bipartite matching [23], planarity testing [22],

* Supported by Research Council of Norway (NFR, no. 274526 and 314528).

and checking the connectivity of a graph [15]. In the field of *parameterized complexity* [7,14], DFS has been instrumental in obtaining *fixed-parameter tractable* (FPT) results by way of *treedepth* [25] and bounded width *tree decompositions* of the given graph [1,16].

In the work reported herein, we refer to the triple (G, r, T), that is, a graph G together with a choice of root vertex r and a DFS tree T, as a *lineal topology* $\mathcal{T}$, or LT for short. This notion of LT corresponds to a point-set topology on the set of edges $E(G)$ (equipped with a rooted DFS tree T), where the open sets are the sets of edges of the subgraphs induced by rooted subtrees with the same root r as T. The lineal topologies of G may differ in terms of the properties of T, such as height and number of leaves. Figure 1 shows one way of representing an LT as a topological graph or drawing in the plane. Given a graph G and a



(a) A graph G (b) $\mathcal{T}_1$ (c) $\mathcal{T}_2$

Fig. 1: A given graph G, and two examples of a lineal topology of G, denoted by $\mathcal{T}_1$ and $\mathcal{T}_2$. They differ in the height and number of leaves of the DFS tree T. The leaves are shown as squares. The tree edges are shown in heavy lines, while the back edges are shown in thin curved lines.

DFS tree (T, r), an embedding of G in the plane so that every pair of edges that cross is a pair of back edges having at most one crossing point is an instance of an LT of G called *T-embedding* [19]. By definition, there exists a T-embedding of the graph G with no crossings points among the back edges if and only if G is a *planar graph*. This is the basis of Hopcroft and Tarjan's linear time planarity testing algorithm [22] and other algorithms for planarity testing, embedding, and Kuratowski subgraph extraction based on de Fraysseix and Rosenstiehl's Left-Right characterization of planarity [8,9].

Considering the above-mentioned applications of LT and the interesting outcomes enabled by the properties of DFS trees, it will be worthwhile to investigate how their structural properties are related to other properties of graphs, including *crossing number* [20] and *bandwidth* [3], useful in algorithms for VLSI design and graph drawing. Then, a complementary study is the complexity of finding those kinds of lineal topologies. We take the first step in this direction of research by investigating the complexity of two main classical decision problems, namely k-Minimum Leafy LT (k-Min-LLT) and k-Maximum Leafy LT (k-Max-

LLT), which correspond to finding an LT with *minimum number of leaves*, and an LT with *maximum number of leaves* respectively.

Given an undirected graph G, the k-Min-LLT and k-Max-LLT problems ask whether G has an LT defined by a DFS tree with at most k and at least k leaves, respectively. One observation that is easy to make is that a Hamiltonian Path (HP) rooted at one of the end vertices of the path defines LT with one leaf. Thus, k-Min-LLT is clearly NP-complete because it is a generalization of the HP problem. To the best of our knowledge, the complexity of k-Max-LLT has not been previously considered. While there are several results regarding the complexity of Maximum Leaf Spanning Tree and Minimum Leaf Spanning Tree in general [2,26,27,28], for DFS trees, the only available complexity results are due to Fellows et al. [17]. For a given graph G, they considered the difficulty of finding a DFS tree that satisfies upper or lower bounds on two parameters, namely $min(G,T)$ and $max(G,T)$, which stand for the minimum length of a root to leaf path of the DFS tree T, and the maximum length of such a path respectively. They showed that, for a given graph G and an integer $k \geq 0$, the following problems are NP-complete: $min(G,T) \leq k$, $min(G,T) \geq k$, $max(G,T) \leq k$, and $max(G,T) \geq k$. It was also shown that, unless $P = NP$, none of these problems admits a polynomial-time absolute approximation algorithm.

Consequently, an appropriate framework within which to study these sorts of problems is parameterized complexity (PC) [14], according to which problems can be analyzed in terms of other parameters apart from the input size. This leads to algorithms for which we pay an exponential cost in the parameter, thereby solving the problem efficiently on instances with small values of the parameter. For the basics of PC necessary to understand this paper, see Section 2.

We consider a parameterization of k-Min-LLT and k-Max-LLT, where the parameter k is the size of the solution (number of leaves), and their so-called "dual" parameterization, namely Dual Min-LLT (*Does G have an LT with at most $n-k$ leaves?*) and Dual Max-LLT (*Does G have an LT with at least $n-k$ leaves?*), where the parameter k is the number of internal vertices. These four parameterized problems are formally defined in Section 2. We show that while each parameterized problem and its parametric dual are trivially the same problem and NP-hard when considered classically, when analyzed in terms of PC, the tractability outcomes differ, with one being FPT and the other W[1]-hard.

1.1 Our Results

Our first result is the hardness of k-Max-LLT. By a reduction from the Multicolored Independent Set (MIS) problem, we show that k-Max-LLT is hard for W[1] parameterized by k. Furthermore, we show that all four problems, considered classically, are NP-hard, and we show trivially that k-Min-LLT parameterized k is para-NP-hard. Our main contribution is in showing the existence of an FPT algorithm for Dual Min-LLT and Dual Max-LLT, parameterized by k, via an application of Courcelle's theorem [4,6], which relates the expressibility of a graph property using *monadic second-order logic* to its tractability

in linear time on graphs with bounded *treewidth* (or *pathwidth*). For a formal definition of this logical language, see Section 2.4.

2 Preliminaries

Unless otherwise specified, a graph G with vertex set $V(G)$ and edge set $E(G)$ is simple, finite, undirected, and connected. For a graph G, n and m denotes the number of vertices $|V(G)|$ and the number of edges $|E(G)|$ of G, respectively. We use uv instead of $\{u, v\}$ to denote an edge in $E(G)$. For any vertex $v \in V(G)$, the set $N_G(v)$ denotes the *open neighborhood* of v, that is, the set of neighbors of v in G, and $N_G[v] = N_G(v) \cup \{v\}$ denotes its *closed neighborhood* in G. We drop the G in the subscript if the graph is clear from the context. Given any two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, if $V_1 \subseteq V_2$ and $E_1 \subseteq E_2$ then G_1 is a *subgraph* of G_2, denoted by $G_1 \subseteq G_2$. If G_1 contains all the edges $uv \in E_2$ with $u, v \in V_1$, then we say G_1 is an *induced subgraph* of G_2, or V_1 induces G_1 in G_2, denoted by $G[V_1]$. If there exists are bijective mapping $f : V_1 \to V_2$ that preserves adjacency, that is, $uv \in E_1$ if and only of $f(u)f(v) \in E_2$, then G_1 is *isomorphic* to G_2 and f is called an *isomorphism*. If G_1 is such that it contains every vertex of G_2, i.e., if $V_1 = V_2$ then G_1 is a *spanning subgraph* of G_2. Given a set of vertices $X \subseteq V$, we express the induced subgraph $G[V(G) \setminus X]$ as $G - X$. If $X = \{x\}$, we write $V(G) \setminus x$ instead of $V(G) \setminus \{x\}$ and $G - x$ instead of $G - \{x\}$. For any pair of vertices $uv \in V(G)$ in a given graph G, we denote any path from u to v by $P(u, v)$, and any path of length ℓ by P^ℓ. A vertex u is said to be *reachable* from a vertex v if there is a path $P(u, v)$. Given a graph G, a set of vertices $S \subseteq V(G)$ is a *connected vertex cover* (CVC) of G if the subgraph $G[S]$ induced by S is connected and S is a *vertex cover* of G, i.e., for every edge $uv \in E(G)$, either $u \in S$ or $v \in S$.

2.1 Lineal Topology

Here, we focus on the definitions of the substructures of lineal topologies that are relevant to our proofs. We refer the reader to [11] for details about basic tree terminologies such as root, parent, child, ancestor, etc. In all cases, a DFS tree is simply denoted by T instead of (T, r) if the root is clear from the context. For any given lineal topology $\mathcal{T} = (G, r, T)$, we denote the height of $\mathcal{T}$, that is, the maximum number of edges in any leaf-to-root path of T, by h. A *leaf* of $\mathcal{T}$ is a vertex that has no descendants but is adjacent to one or more ancestors with respect to T (see Figure 1). We denote by Y and X the set of leaves and *internal vertices* of $\mathcal{T}$, respectively. Given a set of vertices $S \subseteq V(G)$, such that the subgraph $G[S]$ induced by S is connected, we denote the DFS tree of $G[S]$ rooted at $x \in S$ by (T_S, x). The set $E(T)$ and $E(B)$ denote the tree edges and back edges of T respectively. By definition, the set Y is an *independent set*; that is, no pair of vertices $uv \in Y$ are adjacent. This is also true for the set of vertices $U_i \subseteq V(T)$ at each level i of T. Given a vertex v, the set P_v denotes the vertices on the uniquely determined path $P(v, r)$ in T.

2.2 Parameterized Complexity

Now we review some important concepts of parameterized complexity (PC) relevant to the work reported herein. For more details about PC, we refer the reader to [7,14]. Let Σ be a fixed, finite alphabet. A *parameterized problem* is a language $P \subseteq \Sigma^* \times \mathbb{N}$. For an instance $(x, k) \in \Sigma^* \times \mathbb{N}$, $k \in \mathbb{N}$ is called the *parameter*. A parameterized problem P is classified as *fixed-parameter tractable* (FPT) if there exists an algorithm that answers the question "$(x, k) \in P$?" in time $f(k) \cdot poly(|x|)$, where f is a computable function $f : \mathbb{N} \to \mathbb{N}$. Let P and P' be two parameterized problems. A *parameterized reduction* from P to P' is an algorithm that, given an instance (x, k) of P, produces an equivalent instance (x', k') of P' such that the following conditions hold:

1. (x, k) is a YES-instance of P if and only if (x', k') is a YES-instance of P'.
2. There exist a computable function $f : \mathbb{N} \to \mathbb{N}$ such that $k' \leq f(k)$.
3. The reduction can be completed in time $f(k) \cdot poly(|x|)$ for some computable function f.

The *W-hierarchy* [13] captures the level of the intractability of hard parameterized problems. For the purpose of the discussions in this paper, it is enough to note that a problem that is hard for $W[1]$ cannot be solved in FPT running time, unless $FPT = W[1]$. A parameterized problem that is already NP-hard for some single fixed parameter value k (such as $k = 3$ for Graph k-Coloring) is said to be *para-NP-hard*.

2.3 Problem Definitions

We formally define the parameterized problems studied in this work as follows:

k-Min-LLT
Input: A connected undirected graph $G = (V, E)$.
Parameter: k
Question: Does G admit an LT with $\leq k$ leaves?

k-Max-LLT
Input: A connected undirected graph $G = (V, E)$
Parameter: k
Question: Does G have an LT with $\geq k$ leaves?

Dual Min-LLT
Input: A connected undirected graph $G = (V, E)$ and positive integer k
Parameter: k
Question: Does G admit an LT with $\leq n - k$ leaves?

Dual Max-LLT
Input: A connected undirected graph $G = (V, E)$ and positive integer k
Parameter: k
Question: Does G have an LT with $\geq n - k$ leaves?

Below, we present the definitions of the concepts used in Section 3, to show that Dual Min-LLT and Dual Max-LLT are FPT with respect to k.

Definition 1 (Treewidth, Pathwidth). *A tree decomposition of a given graph G is a pair (T_D, B) where T_D is a tree and B is a family of subsets $\{B_i \subseteq V(G) : i \in V(T_D)\}$, called bags, with each node in T_D associated with a bag, satisfying the following properties: (1) $\bigcup_{i \in V(T_D)} B_i = V(G)$, (2) $\forall_{uv \in E(G)}, \exists i \in V(T_D) : u \in B_i$, and $v \in B_i$, and (3) $\forall_{v \in V(G)}$, the set $T_D^i = \{i \in V(T_D) : v \in B_i\}$ gives rise to a connected subtree of T_D.*

The width of (T_D, B) is $max\{|B_i| : i \in V(T_D)\} - 1$ and the treewidth of a graph, denoted $tw(G)$, is the minimum width over all tree decompositions of G. If T_D is a path, then (T_D, B) is called the path decomposition of G and the minimum width over all path decompositions of G is its pathwidth, often denoted by $pw(G)$. For any graph G, it is a fact that $tw(G) \leq pw(G)$ [25].

2.4 Logic of Graphs

We now introduce the basic definitions and notations of the logic of graphs and MSO, the logical language with which we specify the properties associated with Dual Min-LLT and Dual Max-LLT in Section 3. For a thorough discussion of these topics, we refer the reader to [6,24].

Recall that *Second-Order Logic* (SO) is an extension of *First-Order Logic* (FO) that allows quantification over predicates or relations of arbitrary arity. *Monadic Second-Order Logic* (MSO) are SO formulas in which only quantification over unary relations (i.e., subsets of the domain) is allowed. To express graph properties using MSO, a graph $G = (V, E)$ can be represented either as a *logical (or relational) structure* $\lfloor G \rfloor$ whose *domain* is the vertex set V, with a binary relation *adj* on V representing the edges, or as a logical structure $\lceil G \rceil$ whose domain is formed by the disjoint union of V and E, with a binary relation *inc* representing the *incidence* between the vertices and edges of G. There are two main variants of MSO: MSO_1 and MSO_2, corresponding to $\lfloor G \rfloor$ and $\lceil G \rceil$ respectively.

Definition 2 (MSO_1 language). *The logical expressions or formulas of this language are built from the following elements:*

1. *Small variables $u, v, x_1, \ldots, x_k$ for vertices*
2. *Big variables $X, Y, U_1, \ldots, U_k$ for sets of vertices.*
3. *Predicates $adj(u, v)$ and $u \in V$ for adjacency and membership respectively, and "=" equality testing.*
4. *The logical connectives $\vee, \wedge, \neg, \Rightarrow$*
5. *$\forall x, \exists x$ for quantification over vertices and $\forall U$, $\exists U$ for quantification over vertex sets.*

The MSO_2 language extends MSO_1 with variables denoting edges and subset of edges, and the predicate $inc(x, e)$ for incidence, and allows quantification over edges and edge sets.

For clarity, we use $\forall_x$, $\exists_x$ instead of $\forall x$, $\exists x$, and $\forall_U$, $\exists_U$ instead of $\exists U$, and $\forall U$. Given a graph G belonging to a class of graphs $\mathcal{C}$ and a formula Φ expressing a graph property in MSO, we denote the statement "*Φ is true of the vertices and the relation of G*" by $G \models \Phi$ (read as "G models Φ" or "Φ holds on G"). For a logical language $\mathcal{L} \in \{MSO_1, MSO_2\}$, we say that a graph property is $\mathcal{L}$-*expressible* if there exists a formula (sentence) of $\mathcal{L}$ for expressing it. The theorem below states the consequence of expressing a graph property by an MSO_2 formula.

Lemma 1 (Courcelle's theorem [4,6]). *Assume that ϕ is a fixed MSO_2 formula of length ℓ expressing a graph property. Then for any graph G belonging to a graph class $\mathcal{C}$ with treewidth bounded by a fixed positive integer k, there is an algorithm that takes G and its tree decomposition as input and decides whether $G \models \phi$ in time $\mathcal{O}\big(f(\ell, k) \cdot n\big)$, for some computable function f.*

By a theorem similar to Lemma 1, every graph property that is MSO_1-expressible can be decided in linear time on graphs of bounded *clique-width*, a graph complexity measure that is similar to treewidth [5]. It is worth noting that Lemma 1 also holds for MSO_1 formulas because every graph property expressible by an MSO_1 formula is also expressible by an MSO_2 formula; but the converse is not true. For example, the existence of a Hamiltonian path can only be expressed in MSO_2 [6]. Therefore, it is stronger to claim that a given property is MSO_1-expressible. In Section 3.2, we show that the property of having an LT with at least $n - k$ leaves or at most $n - k$ leaves is MSO_1-expressible.

3 Complexity Analysis

In this section, we consider the four natural parameterized problems defined above. We first show that k-Max-LLT is hard for W[1] parameterized by k. The reduction in the proof is polynomial and thus the problem is NP-hard. Besides, we show trivially that $k-$Min-LLT is para-NP-hard with respect to k. This is followed by proofs of hardness for Dual Min-LLT and Dual Max-LLT, considered classically. Moreover, we show that these two problems are FPT with respect to k. To this end, we construct MSO_1 formulas ϕ_k and φ_k to express the property of having an LT with at most $n - k$ leaves and that of having an LT with at least $n - k$ leaves. Next, we make use of the following facts on the height of a lineal topology to show the existence of an FPT algorithm for the two problems. Given a graph G and an integer $k \geq 0$, we show, for Dual Max-LLT, that if the height of the DFS tree T resulting from any DFS of G is more than $2^{k+1} - 2$, then T witnesses that the answer is NO, otherwise, G has a path decomposition of width at most $2^{k+1} - 1$. For Dual Min-LLT, we trivially show that the answer is YES if the number of internal vertices of T is at least k, otherwise, G has a path decomposition of width at most k.

3.1 Hardness Results

Theorem 1. *k-Max-LLT is W[1]-hard parameterized by k and NP-complete when considered classically.*

Proof. We reduce from the parameterized Multicolored Independent Set (MIS) problem parameterized by the number of colors. In the MIS problem, we are given a graph $G = (V, E)$ and a coloring of V with k colors, and the task is to determine whether G has a *k-colored independent set*, that is, a k-sized independent set containing one vertex from each color class. We trivially assume that each color class induces a clique for our argumentation. This problem is W[1]-hard with respect to k [7, chapter 13], which implies that it is unlikely that it can be solved in time $f(k) \cdot poly(n)$ for any computable function f.

Given a positive integer k, let G be an instance of the MIS problem in which $\{V_1, \ldots, V_k\}$ is a partition of the vertex set $V(G)$ such that, for each $i \in [1, k]$, V_i induces a clique and corresponds to a color class. Now we construct an instance (G', k) of the k-Max-LLT problem from G by introducing a set of k universal vertices $U = \{u_1, u_2, \ldots, u_k\}$, i.e., every $u_i \in U$ is adjacent to every vertex in G and in $U \setminus u_i$. The completed $G' = (V', E')$ has $V' = V \cup U$, and $E' = E(G) \cup \{u_i v \mid u_i \in U, v \in V(G') \setminus u_i\}$ (see Figure 2). The main idea of this construction, as will be argued below, is to enable a depth-first traversal of G that guarantees an LT with at least k leaves corresponding to a k-sized independent set in G, if it exists. It is not hard to see that we can construct (G', k) from (G, k) in polynomial time. Lemmas 2 and 3 below show that G' admits an LT with at least k leaves if and only if G has a k-colored independent set.

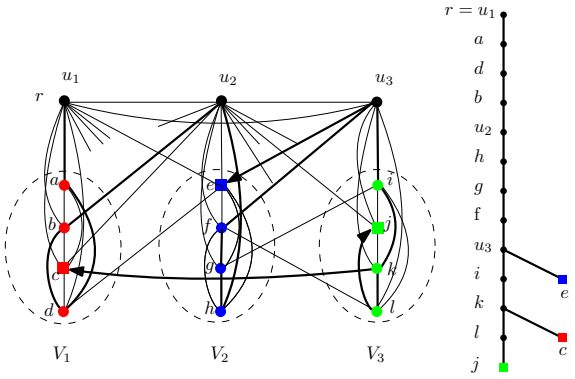


Fig. 2: An example of a reduction from an instance G of MIS, with k-colored independent set $X = \{c, e, j\}$, to an instance G' of Max-LLT and a DFS of G' that yields a DFS tree T' with $\{c, e, j\}$ as its leaves.

Lemma 2. *If a k-colored independent set exists in G, then G' admits an LT with at least k leaves.*

Proof. Suppose that $X = \{x_1, ..., x_k\}$ is a k-colored independent set in G. Since $x_1 \in V_1, \ldots, x_k \in V_k$ with each color class V_i inducing a clique, any depth-first traversal of G' that excludes the vertices in X until all the vertices in $V(G') \setminus X$ have been visited yields an LT with the vertices in X as its leaves. One way to achieve this is to start from the vertex $u_1 \in U$ and visit every vertex in the corresponding color class V_1 except x_1. Next, choose $u_2 \in U$ and explore every vertex in V_2 except x_2. Repeat this process sequentially for each $u_i \in U$ and its corresponding color class V_i until the last vertex $u_k \in U$ is reached. Now choose the vertex x_k after exploring every vertex in the set $V_k \setminus \{x_k\}$. See Figure 2 for an illustration of this process. At this point, every edge incident to x_k leads to a vertex already reached by the DFS because X is an independent set. Thus, x_k becomes a leaf in the resulting DFS tree T' of G'. If any vertex $x_i \in X$ is adjacent to an already visited vertex $v \in V_k$, we backtrack and choose x_i from v. Otherwise, we backtrack to $u_k \in U$, as every vertex in X is reachable from this vertex by construction. Each of the remaining vertices $\{x_1, \ldots, x_{k-1}\}$ reached by DFS becomes a leaf in T' because of the same reason as for x_k. □

Lemma 3. *If an LT with at least k leaves in G' exists, then there is a k-colored independent set in G.*

Proof. If $k = 1$ then G is obviously a YES-instance. Suppose that $k \geq 2$ and G' admits an LT in which $X = \{x_1, x_2, ..., x_k\}$ are the leaves. Observe that X is an independent set. Then, based on the following claims, we conclude that X induces a k-colored independent set in G.

Claim 3.1. Each color class V_i in G' can contain at most one vertex from X.

Proof. The set of leaves X is an independent set, and by construction, each color class V_i in G' is a clique. Therefore, there cannot be any LT of G' with two or more leaves from the same color class. ⧫

Claim 3.2. None of the vertices in X is from the vertex set $U = \{u_1, \ldots, u_k\}$.

Proof. For $i \in [1, k]$, if u_i is a leaf of T', the remaining vertices in $V(G') \setminus u_i$ must necessarily be internal vertices of T' by construction. Since X contains at least 2 vertices, it follows that no vertex in U can be in X. ⧫

Combining Claims 3.1 and 3.2, we conclude that X is a k-colored independent set in G.

□

Theorem 2. k-Max-LLT *is NP-complete.*

Proof. k-Max-LLT is clearly in NP. The NP-hardness of the problem follows from the proof of Theorem 1, because Multicolored Independent Set is NP-complete [7,20] and the reduction from Multicolored Independent Set to k-Max-LLT is a polynomial-time reduction.

Theorem 3. k-Min-LLT *parameterized by k is para-NP-hard.*

Proof. This follows, trivially, from the fact that k-Min-LLT is NP-hard already for $k = 1$, because for this case, k-Min-LLT is equivelent to Hamiltonian Path which is well-known to be NP-complete [20].

Theorem 4. Dual Min-LLT *and* Dual Max-LLT*, considered classically, are NP-hard.*

Proof. Dual Min-LLT can be restricted to the Hamiltonian Path problem by allowing only instances in which $k = n - 1$. Similarly, $(G, n - k)$ is a YES-instance of k-Max-LLT iff (G, k) is a YES-instance of Dual Max-LLT. □

3.2 MSO Formulations

Recall that a DFS tree T is a tree formed by a set of edges $E(T)$ with a choice of a root vertex r, such that every edge not in T connects a pair of vertices that are related to each other as an ancestor and descendant in T. For any DFS tree T of a given graph G, the set of vertices U_i at each level i of T is an independent set of G. Thus, T corresponds to a partition of the vertex set $V(G)$ into a sequence of independent sets $(U_0, U_1, ..., U_h)$ with $h \in \mathbb{N}$, where the root r is the only member of U_0 and U_h contains the vertex witnessing the height h of T (see Figure 3a). Based on this observation, we provide the following definition of a

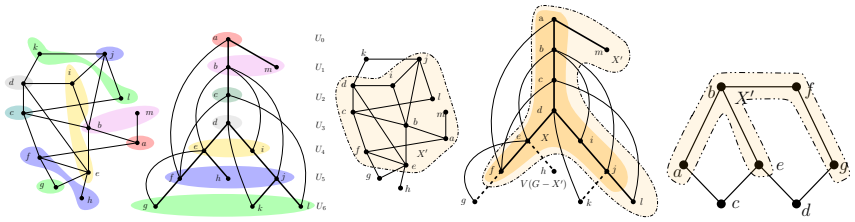


Fig. 3: (a) Color classes denoting an LT partition $(U_0, U_1, ..., U_6)$ of a graph G with $n = 13$, and a representation of the associated LT of G with height $h = 6$ and number of internal vertices $|X| = 8$. (b) A CVC X' of a graph G, and a DFS tree T of G formed by a DFS tree $(T_{X'}, a)$ for $G[X']$ and the independent set $V(G - X')$. The internal vertices X of T consist of $\{a, b, c, d, e, f, i, j\}$. Thus $|X| \leq |X'|$. (c) A CVC X' of a graph G such that $G[X']$ does not admit a DFS tree $T_{X'}$ which extends to a DFS tree T of G.

DFS tree with bounded height, which allows us to express the properties "G has an LT with at least $n - k$ leaves" and "G has an LT with at most $n - k$ leaves" in MSO_1.

Definition 3. *Let G be a graph and h a positive integer. A tree-partition of G of height h is a sequence $(U_0, \ldots, U_h)$ with $U_0, \ldots, U_h \subseteq V(G)$ such that:*

1. $(U_0, U_1, \ldots, U_h)$ *is a partition of* $V(G)$.
2. U_0 *contains only one element* r.
3. *Every vertex* $u \in U_i$ *has a unique neighbor* $v \in U_{i-1}$ *for all* $i \in [1, h]$.

The tree associated with the tree-partition $(U_0, \ldots, U_h)$ *is the rooted spanning tree* T *of* G *with root* r *and edge set* $E(T) = \{uv : uv \in E(G), u \in U_i, v \in U_{i-1}$ *for all* $i \in [1, h]\}$. *We say that the tree-partition* $(U_0, \ldots, U_h)$ *is an LT-partition of* G *if:*

4. *For every edge* uv *of* G, u *is an ancestor of* v *in* T *or* v *is an ancestor of* u *in* T.

Lemma 4. *For every* $h \in N$, *a graph* $G = (V, E)$ *has an LT with height* h *if and only if* G *admits an LT-partition* $(U_0, U_1, ..., U_h)$.

Proof. Let T be the spanning tree of G associated with $(U_0, U_1, ..., U_h)$. Property 4 of Definition 3 ensure that every $uv \in E \setminus E(T)$ is a back edge, and, thus, for all $i \in [0, h]$, U_i is an independent set. This gives us an LT with height h.

Conversely, suppose we have an LT of G defined by a DFS tree T with height h. Then it is easy to see that the root and the vertices at each level of T constitute an LT-partition $(U_0, U_1, ..., U_h)$. □

Dual Max-LLT. Let G be a graph and $k \in \mathbb{N}$. If $k = 1$ then G is a YES-instance of Dual Max-LLT if and only if G is a *star*. Thus, in what follows, we assume that $k \geq 2$ and $|G| \geq 3$. Suppose that G admits an LT with at least $n-k$ leaves, and consider a DFS tree (T, r) witnessing that G is a YES-instance. We can readily observe that the internal vertices X of (T, r) is a CVC of G, and the subtree (T_X, r) with height $h \leq |X|$ is a DFS tree of the graph $G[X]$ induced by X. This is also true for any subtree $(T_{X'}, r)$, such that $X' \supseteq X$ and $|X'| \leq k$ (see Figure 3b). However, given a graph $G = (V, E)$ and a CVC X of G, $G[X]$ may not have a DFS tree T_X that can be extended to a DFS tree T of G by adding the vertices in $V(G - X)$ to T_X as leaves. An example of such a CVC of a given graph is shown in Figure 3c. Using these intuitive ideas, we characterize the graphs that admit an LT with at least $n - k$ leaves by Lemma 5. See the full version of this paper for the proof of this lemma.

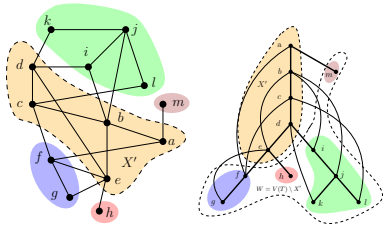
Lemma 5. *A graph* G *has an LT with at least* $n - k$ *leaves if and only if it has a set of vertices* X' *of size at most* k *satisfying the following properties:*

1. X' *is a connected vertex cover of* G.
2. $G[X']$ *admits an LT partition* $(U_0, ..., U_h)$ *with* $h \leq |X'|$ *such that, for any vertex* $y \in V(G) \setminus X'$, *if* y *is adjacent to a pair of vertices* $u, v \in X'$, *then either* u *is the ancestor of* v *or* v *is the ancestor of* u *in the LT formed by* $(U_0, ..., U_h)$.

Theorem 5. *For all* $k \in N$, *there exists an* MSO_1 *formula* ϕ_k *such that for every graph* G, *it holds that* $\lfloor G \rfloor \models \phi_k$ *iff* G *is a YES-instance of* Dual Max-LLT.

To prove this, we construct the formula ϕ_k as a conjunction of formulas expressing the properties in Definition 3 and Lemma 5; see the full version of this paper.

Dual Min-LLT For $k = 1$, every graph G is a YES-instance of Dual Min-LLT. Thus, henceforth, we assume that $k \geq 2$ and $|G| \geq 3$. Let G be a given graph that admits an r-rooted LT with at most $n - k$ leaves (i.e., at least k internal vertices), and consider a DFS tree T of G witnessing this fact. Then any subtree $(T_{X'}, r)$ of T, where $X' = \{x_1, \ldots, x_k\}$ is a set of k internal vertices and $r \in X'$ is the same as the root of T, is a DFS tree for the subgraph $G[X']$ induced by X'; see Figure 4 for an example. Observe that, every leaf of $(T_{X'}, r)$



(a) (b)

Fig. 4: (a) A graph G with a connected subset of vertices X' of size 5, and (b) a DFS tree $(T_{X'}, a)$ for $G[X']$ that extends to an LT of G with internal vertices $X' \cup \{f, i, j\}$. W consists of three maximal connected subgraphs of G shown in different colors.

is adjacent to a vertex in the set $W = V(T) \setminus X'$, and each of the subtrees that extend $T_{X'}$ to form T is a DFS tree for some maximal connected component of $G - X'$.

As a result, we transform the problem of determining whether G admits an LT with at most $n-k$ leaves to that of deciding whether there exists a subset of k vertices X', such that, the subgraph $G[X']$ admits an r-rooted DFS tree $T_{X'}$ isomorphic to a subtree (T', r) of an LT of G witnessing that G is a YES-instance. To this end, we introduce the following definition.

Definition 4. *Let G be a graph and $X' = \{x_1, \ldots, x_k\}$, a set of k vertices that induces a connected subgraph of G. We say that a tree-partition $(U_0, \ldots, U_h)$ of $G[X']$ of height $h \leq k$ is a partial LT-partition of G or extends to an LT-partition of G if it satisfies the following property: for every $W' \subseteq V(G)$ such that $G[W']$ is a maximal connected subgraph of $G - X'$, there exists $x' \in X'$ such that any vertex in $x \in X'$ with at least one neighbor in W' is an ancestor of x'.*

In Figure 4, it is easy to see that the subgraph induced by $\{k, j, i, l\}$ does not have a tree-partition that forms a partial LT-partition of G. For a partial LT-partition of a given subgraph $G[X']$ of size k to yield an LT of G with at most

$n - k$ leaves, it is necessary that every leaf of the partial LT is adjacent to at least one vertex in $V(G) \setminus X'$. Based on these intuitive ideas, we characterize the YES-instances of Dual Min-LLT by Lemma 6; see the full version of this paper for the proof.

Lemma 6. *For every $k \in N$, a graph $G = (V, E)$ has an LT with the number of leaves $\leq n - k$ if and only if there exists a set of vertices X' of size at least k satisfying the following property: $G[X']$ admits a partial LT-partition $(U_0, \ldots, U_h)$ of height $h \leq k$ in which every leaf is adjacent to at least one vertex in $W = V(G) \setminus X'$.*

Now, we use Lemma 6 to obtain Theorem 6; see the full version of this paper for the proof.

Theorem 6. *For all $k \in N$, there exists an MSO_1 formula ψ_k such that for every graph G, it holds that $\lfloor G \rfloor \models \psi_k$ iff G is a YES-instance of* Dual Min-LLT.

3.3 FPT Algorithms for Dual Min-LLT and Dual Max-LLT

In this section, we show the existence of an FPT algorithm for Dual Min-LLT and Dual Max-LLT using the MSO formulations above, and Lemmas 7 and 8. For the proofs of these lemmas, see the full version of this paper.

Lemma 7. *Given a graph G and a positive integer t, if G admits an LT with height at most t, then G has a path decomposition of width at most t that can be computed in linear time.*

Lemma 8. *Given a graph G and a positive integer k, if G admits an LT of height at most k, then the length of any path in G is at most $2^{k+1} - 2$.*

Theorem 7. Dual Max-LLT *parameterized by $k \in \mathbb{N}$ is in FPT.*

Proof. Let G be a graph and k a positive integer. One observation that is easy to make is that if G is a YES-instance of Dual Max-LLT, then G admits an LT with height at most k. Thus, Dual Max-LLT can be solved as follows: (1) Construct a DFS tree T by performing a DFS of G. If the height h of T is more than $2^{k+1} - 2$, then we know, by Lemma 8 that G does not admit an LT with $h \leq k$. Therefore, return NO and stop. (2) Otherwise, use Lemma 7 to read off a path decomposition of G of width at most $2^{k+1} - 1$ from T, one bag per leaf. (3) Applying Courcelle's theorem with Theorem 5 and this path decomposition, it follows that checking whether G is a YES-instance is FPT in k. □

Theorem 8. Dual Min-LLT *parameterized by $k \in \mathbb{N}$ is in FPT.*

Proof. The proof follows steps analogous to that of Theorem 7. Construct any DFS tree T. If T has at most $n - k$ leaves, return YES and stop. Otherwise, we use Lemma 7, to obtain a path decomposition of G of width at most k from T. With this, Theorem 6, and Courcelle's theorem implies that we can derive an FPT algorithm that runs in time linear in n to check whether G is a YES-instance. □

4 Conclusion

In this paper, we have shown hardness results for four natural parameterized problems that have to do with finding an LT (or DFS tree) with a restricted number of leaves. Our theorem shows that k-Max-LLT is hard for W[1]. This raises the natural question of where it belongs in the W-hierarchy with respect to membership. Is it in W[1] and thus W[1]-complete? Is it in W[P]? It seems to be AND-compositional, like the Bandwidth problem, as discussed in [18] and we conjecture that k-Max-LLT cannot be in W[P] for reasons similar to the case of Bandwidth. We have also shown that Dual Min-LLT and Dual Max-LLT are FPT parameterized by k. Instead of relying on Courcelle's theorem to show the existence of an FPT algorithm for these problems, we believe it should be possible to construct an algorithm that solves each problem explicitly via dynamic programming over the path decomposition returned by our algorithm. An obvious question is whether both problems admit a polynomial kernel.

On the complexity of finding an LT with restricted height h, the only known results are the NP-hardness results due to Fellows et al. [17] (see Section 1). A consequence of our Theorem 7 is that the problem $h \leq k$ is FPT with respect to k. The natural parameterized problem $h \geq n - k$ is para-NP-complete since it is equivalent to Hamiltonian Path when $k = 1$. We plan to investigate the PC of: (i) $h \leq n - k$, and (ii) $h \geq k$, which we believe is FPT parameterized by k.

Acknowledgements We acknowledge support from the Research Council of Norway (NFR, no. 274526 and 314528). We also thank Nello Blaser and Benjamin Bergougnoux for helping to review the initial versions of the paper.

References

1. Bodlaender, H.: On linear time minor tests with depth-first search. Journal of Algorithms **14**(1), 1–23 (1993)
2. Bonsma, P.S., Brueggemann, T., Woeginger, G.J.: A faster fpt algorithm for finding spanning trees with many leaves. In: Rovan, B., Vojtáš, P. (eds.) Mathematical Foundations of Computer Science 2003. pp. 259–268. Springer Berlin Heidelberg, Berlin, Heidelberg (2003)
3. Chinn, P.Z., Chvátalová, J., Dewdney, A.K., Gibbs, N.E.: The bandwidth problem for graphs and matrices—a survey. Journal of Graph Theory **6**(3), 223–254 (1982)
4. Courcelle, B.: The monadic second-order logic of graphs. i. recognizable sets of finite graphs. Information and Computation **85**(1), 12–75 (1990)
5. Courcelle, B., Makowsky, J.A., Rotics, U.: Linear time solvable optimization problems on graphs of bounded clique-width. Theory of Computing Systems **33**(2), 125–150 (2000)
6. Courcelle, P.B., Engelfriet, D.J.: Graph Structure and Monadic Second-Order Logic: A Language-Theoretic Approach. Cambridge University Press, USA, 1st edn. (2012)
7. Cygan, M., Fomin, F.V., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: Parameterized Algorithms. Springer Publishing Company, Incorporated, 1st edn. (2015)

8. De Fraysseix, H.: Trémaux trees and planarity. Electronic Notes in Discrete Mathematics **31**, 169–180 (2008)
9. De Fraysseix, H., De Mendez, P.O., Rosenstiehl, P.: Trémaux trees and planarity. International Journal of Foundations of Computer Science **17**(05), 1017–1029 (2006)
10. de Fraysseix, H., Ossona de Mendez, P.: Trémaux trees and planarity. European Journal of Combinatorics **33**(3), 279–293 (2012), topological and Geometric Graph Theory
11. Diestel, R.: Graph Theory. Springer Publishing Company, Incorporated, 5th edn. (2017)
12. Diestel, R., Leader, I.: Normal spanning trees, aronszajn trees and excluded minors. Journal of the London Mathematical Society **63**(1), 16–32 (2001)
13. Downey, R.G., Fellows, M.R.: Fixed-parameter tractability and completeness i: Basic results. SIAM Journal on Computing **24**(4), 873–921 (1995)
14. Downey, R.G., Fellows, M.R.: Fundamentals of Parameterized Complexity. Springer Publishing Company, Incorporated (2013)
15. Even, S., Tarjan, R.E.: Network flow and testing graph connectivity. SIAM Journal on Computing **4**(4), 507–518 (1975)
16. Fellows, M.R., Langston, M.A.: On search decision and the efficiency of polynomial-time algorithms. In: Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing. p. 501–512. STOC '89, Association for Computing Machinery, New York, NY, USA (1989). https://doi.org/10.1145/73007.73055
17. Fellows, M.R., Friesen, D.K., Langston, M.A.: On finding optimal and near-optimal lineal spanning trees. Algorithmica **3**(1–4), 549–560 (Nov 1988)
18. Fellows, M.R., Rosamond, F.A.: Collaborating with Hans: Some Remaining Wonderments, pp. 7–17. Springer International Publishing, Cham (2020)
19. de Fraysseix, H., Rosenstiehl, P.: A characterization of planar graphs by trémaux orders. Combinatorica **5**(2), 9 (1985). https://doi.org/10.1007/bf02579375
20. Garey, M.R., Johnson, D.S.: Crossing number is np-complete. SIAM Journal on Algebraic Discrete Methods **4**(3), 312–316 (1983). https://doi.org/10.1137/0604033
21. Hopcroft, J., Tarjan, R.: Algorithm 447: Efficient algorithms for graph manipulation. Communications of the ACM **16**(6), 372–378 (Jun 1973)
22. Hopcroft, J., Tarjan, R.: Efficient planarity testing. Journal of the ACM (JACM) **21**(4), 549–568 (1974)
23. Hopcroft, J.E., Karp, R.M.: An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. SIAM Journal on Computing **2**(4), 225–231 (1973)
24. Libkin, L.: Elements of Finite Model Theory. Springer (2004)
25. Nešetřil, J., de Mendez, P.O.: Bounded Height Trees and Tree-Depth, pp. 115–144. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
26. Ozeki, K., Yamashita, T.: Spanning trees: A survey. Graphs and Combinatorics **27**, 1–26 (2011)
27. Prieto, E., Sloper, C.: Either/or: Using vertex cover structure in designing fpt-algorithms — the case of k-internal spanning tree. In: Dehne, F., Sack, J.R., Smid, M. (eds.) Algorithms and Data Structures. pp. 474–483. Springer Berlin Heidelberg, Berlin, Heidelberg (2003)
28. Rosamond, F.: Max Leaf Spanning Tree, pp. 1211–1215. Springer New York, New York, NY (2016). https://doi.org/10.1007/978-1-4939-2864-4_228
29. Williamson, S.: Combinatorics for Computer Science. Dover books on mathematics, Dover Publications (2002), https://books.google.no/books?id=YMIoy5JwdHMC